

Simulation of Underwater RF Wireless Sensor Networks using Castalia

Sergio Valcarcel Macua, Santiago Zazo
Javier Zazo, Marina Pérez Jiménez
Universidad Politécnica de Madrid,
Madrid 28040, Spain.

Email: {sergio, santiago}@gaps.ssr.upm.es,
javier.zazo.ruiz@upm.es, marina.perez@isom.upm.es

Iván Pérez-Álvarez, Eugenio Jiménez
Instituto para el Desarrollo Tecnológico y la
Innovación en Comunicaciones (IDeTIC),
Universidad de Las Palmas de Gran Canaria,
Las Palmas, 35017, Spain.

Email: {ivan.perez, eugenio.jimenez}@ulpgc.es

Joaquín Hernández Brito
Plataforma Oceánica de
Canarias (PLOCAN),
Telde 35200, Spain
Email: joaquin.brito@plocan.eu

Abstract—We use real measurements of the underwater channel to simulate a whole underwater RF wireless sensor networks, including propagation impairments (e.g., noise, interferences), radio hardware (e.g., modulation scheme, bandwidth, transmit power), hardware limitations (e.g., clock drift, transmission buffer) and complete MAC and routing protocols. The results should be useful for designing centralized and distributed algorithms for applications like monitoring, event detection, localization and aid to navigation. We also explain the changes that have to be done to Castalia in order to perform the simulations.

I. INTRODUCTION

Underwater sensor networks are useful for environmental monitoring and security applications. While acoustic or optical methods are preferred in deep sea water scenarios, they suffer from several impairments in shallow water settings. In particular, acoustic signals suffer from multi-path propagation, reverberation and ambient noise and, very importantly, they have a negative impact on marine life [1], [2]; while optical signals suffer from high absorption and strong backscatter [3]. RF communications seems an attractive alternative that could offer higher bandwidth and better transmission in medium boundaries (e.g., water–air, seabed–ice). Indeed, we have been able to measure the frequency response and achieve stable links at some meters distance [4], [5]. Now, we want to develop underwater wireless sensor networks (U-WSN) for applications like localization, aid to navigation, event detection and environment monitoring. We consider both centralized and distributed algorithms. In a centralized algorithm, nodes sense the environment and send their measurements to a *sink* node, which is in charge of gathering and processing all data from every node. Distributed algorithms are those in which nodes communicate with their neighbors and exchange some information (not necessarily the measurements but some intermediate variables like their estimate about some magnitude), so that they can predict and interact with the environment. Examples of distributed algorithms include feature extraction for data compression [6], diffusion adaptation [7] and belief propagation algorithms [8]. From a communications point of view, centralized algorithms require routing any packet from any node to the sink, which requires route discovery; while distributed algorithms require point to point communication between neighbors, which may require neighborhood discovery.

Deploying U-WSN presents several challenges, like communication impairments, unexpected and asynchronous events, limited battery life, etc. Thus, we approach the problem in four main stages: *i*) channel characterization, *i*) simulation of U-WSN for specific scenario, *iii*) algorithm design and *iv*) network deployment. Stage 1

includes measurement campaigns to obtain the frequency response and the underwater propagation model underwater. In this paper we focus on stage 2, using the underwater channel characterization explained in the companion papers [4], [5]. In particular, the main contribution of this work is to simulate a whole U-WSN, including propagation impairments (e.g., noise, interferences), radio hardware (e.g., modulation scheme, bandwidth, transmit power), hardware limitations (e.g., clock drift, transmission buffer), complete MAC and routing protocols for studying the influence of different parameters on some standard scenarios. The results of this work should be input to stage 3 for studying the feasibility of some algorithms (e.g., the order of loss packet rate that must be tolerated by the distributed algorithms presented in the companion paper [9]). Stage 4 includes prototype development and building a testbed. The idea is to iterate over these stages in order to minimize the cost of a real deployment.

Castalia [10] is a simulator for Wireless Sensor Networks (WSN) based on the OMNeT++ platform [11] that offers realistic wireless channel and radio models and realistic node behaviour. The main reasons for choosing Castalia are its level of realism, speed and flexibility. The speed is achieved because all the modules are written in C++. The flexibility is at the cost of having no GUI. Indeed, Castalia is a command line simulator, where scenarios and settings are defined through external configuration files. The results are also given in text files, but they can be accessed with convenient parser scripts. In the following sections we explain how we have used and extended Castalia for simulating a U-WSN in some scenarios.

II. WIRELESS CHANNEL

A. Channel model characterization

We start from the measurements of the underwater channel—taken with loop antennas—presented in the companion papers [4], [5]. Figure 1-top shows the measured (solid) and simulated (dashed) channel frequency response. Figure 1-bottom shows the fading of the signal at 46kHz along almost 2 hours.

We have used the measurements of the frequency response to propose a path loss model, where the log of the attenuation is linear with the distance. Let $L(d)$ denote the path loss (dB) at d meters from the transmitter. Then, the proposed model expresses $L(d)$ as an affine function of the distance with parameter η :

$$L(d) = L(d_0) + \eta \frac{d}{d_0} + X \quad (1)$$

where d_0 is some reference distance and X is a random variable.

Note that (1) is different from the standard free-space path loss model, in which the log of the attenuation is linear with the log of

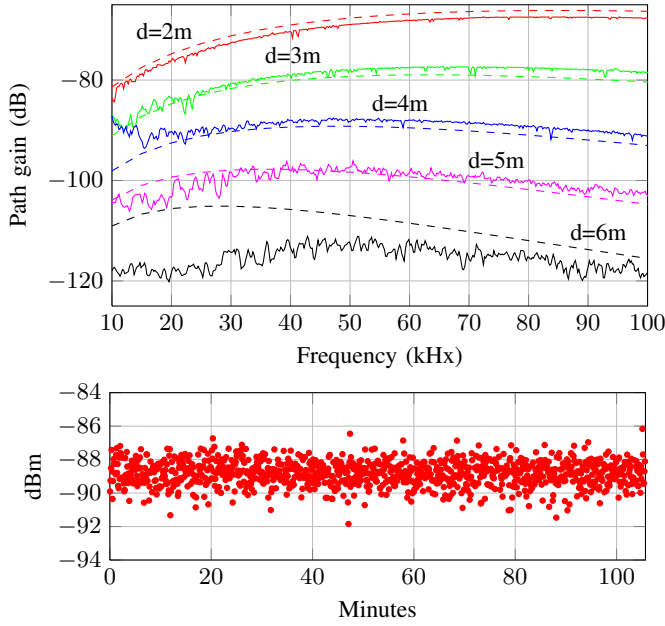


Fig. 1. Measurements of the underwater wireless channel. Frequency response at multiple distances (top). Temporal variation at 46kHz (bottom).

the distance:

$$L_{fs}(d) = L(d_0) + \eta \cdot 10 \cdot \log\left(\frac{d}{d_0}\right) + X \quad (2)$$

We remark that these path loss models should be frequency dependent, but we have assumed the simpler narrow band case.

In order to find parameters $L(d_0)$ and η for both models, we have considered carrier frequency 46kHz and $d_0 = 2$. The vector of distances is $\mathbf{d} = [2, 3, 4, 5, 6]^T$ and the corresponding losses are $\mathbf{m} = [78.5, 88, 98, 112]^T$ (note that the loss is opposite sign to the gain displayed in Figure 1). Let $\mathbf{x} = [\eta, L(d_0)]^T$ be the vector of unknown parameters. Now, for the linear model, we build a new vector $\mathbf{b} = \mathbf{d}/d_0$ and introduce the matrix $\mathbf{A} = [\mathbf{b}, \mathbf{1}]$ of size 6×2 , where $\mathbf{1}$ is a vector of ones. Thus, we have a system of equations, $\mathbf{m} = \mathbf{A}\mathbf{x}$, that we can solve as follows: $\mathbf{x} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{m}$, obtaining linear coefficient $\eta/d_0 = 10.45$ and offset $L(d_0) = 47.40$ dB. The procedure for finding the parameters of the free-space model is the same as for the linear model but replacing \mathbf{b} by the scaled log of the distance, i.e., $\mathbf{b} = 10 \cdot \log(\mathbf{d}/d_0)$.

Figure 2 shows the real measurements at 46kHz (black dots), the proposed linear model (dashed red) and the free-space model (solid blue). It is apparent that the proposed linear model (1) fits the measurements much better than the free-space model (2).

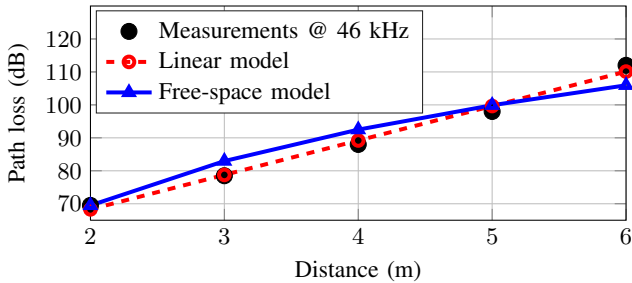


Fig. 2. Measurements and path loss models.

The additive noise X can be also estimated from the trace of measurements displayed in Figure 1-bottom. Figure 3 shows the histogram for such trace together with a normal distribution fit. We conclude that the Normal distribution is a reasonable approximation, so that we can assume $X \sim \mathcal{N}(0, \sigma^2)$, with $\sigma^2 = 0.56$ dB.

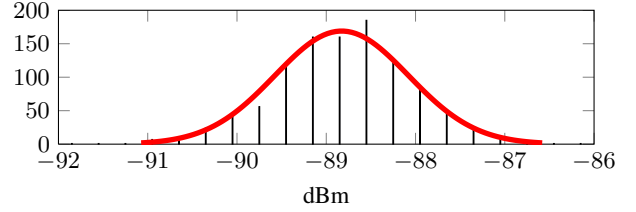


Fig. 3. Histogram and normal distribution fit for the trace of measurements.

A final observation from Figure 1-bottom relevant to the wireless channel simulation is that we can dismiss slow fading.

B. Simulation of the linear channel model

Castalia offers only one parametric channel out of the box, which is the free-space model. Nevertheless, it also offers the possibility of expressing non-parametric path loss models by describing the path loss for every pair of nodes in an external file. For instance, suppose we have a network of N nodes, then, for every node n , we can express the path loss seen by every node $m \neq n$ when node n is transmitting. Although flexible, this approach requires to rewrite the path loss file every time we change the network topology or the model parameters. Hence, we preferred to extend Castalia by including the linear model (1) into the class **WirelessChannel**.

For the temporal variation of the channel gain, Castalia only offers the possibility of writing an external file describing the fading probability distribution at every time instant, allowing to include temporal correlation. Again, although flexible, the main handicap of using an external files is that we have to rewrite them every time we change a parameter (e.g., the variance of the PDF). In our particular case, Figure 3 shows that a simple additive Gaussian random variable fits well the measurements. Hence, we have found more convenient to extend Castalia by adding a normally distributed fast fading model to the class **channelTemporalModel**.

III. RADIO PARAMETERS

We have chosen an FSK modulation scheme since it is optimal in the low signal-to-noise-ratio (SNR) regime. Castalia offers FSK with noncoherent receiver and no channel coding. We have added coherent receiver and channel coding to the class **Radio** (in particular, to the method **Radio::SNR2BER**). We have selected a convolutional code (3, 1, 2), whose nonsystematic feedforward encoder is represented by $G(D) = [1 + D, 1 + D^2, 1 + D + D^2]$, with free distance $d_{free} = 7$ and $B_{d_{free}} = 1$ (this is the total number of nonzero information bits on all weight- d_{free} paths, divided by the number of information bits per unit time). Let p be the BER obtained from the SNR curve for our FSK modulation. We have characterized the system by including the coding gain as the BER upper bound given by [12, Eq. 12.1]:

$$\text{BER}(p, B_{d_{free}}, d_{free}) \approx B_{d_{free}} \left[2\sqrt{p(1-p)} \right]^{d_{free}} \quad (3)$$

The radio parameters common to all simulations are: data rate 3kbps, 1 bit per symbol, signal bandwidth 12kHz, noise bandwidth 12kHz, noise floor -108dBm and receiver sensitivity -101dBm.

The required transmit power depends on the scenario under study and influences critically on the communication range. Section IV-A

shows simulation results for 10dBm, 20dBm and 30dBm, while Sections IV-B–IV-D use 30dBm. Castalia has a prefixed value of 0dBm as the maximum power allowed by the simulator, which must be updated for our setting. In particular, it is needed to increase the value of variable `maxTxPower` in class `WirelessChannel`.

The maximum packet size depends on the application under study. For instance, the localization algorithm considered in the companion paper [9] requires small packets of size 8 bytes (i.e., 2 float numbers per iteration) before encoding. Recall that the larger the packet, the higher the probability that it suffers some impairment. Section IV-A shows simulation results for random packets of 30 bytes, while the rest of simulations transmit their neighbor list, which consists of around 15 bytes.

IV. SCENARIOS UNDER TEST

A. Point to point

The first scenario under study is a 2-nodes, point-to-point setting, where one node transmits and the other listens (see Figure 4). We

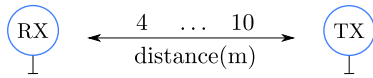


Fig. 4. Point to point scenario with 2 nodes.

study the successfully received packet rate for different distances between 4 and 10 meters. Figure 5 shows results (averaged over 20 independent trials) for 3 different transmit power: 10dBm (red dotted), 20dBm (black dashed) and 30dBm (blue solid); and for 4 different receivers: noncoherent with no channel coding (square), coherent with no channel coding (circle), noncoherent with channel coding (triangle) and coherent with channel coding (no marker). As expected, including a coherent receiver and some channel coding improves the rate. We conclude that with the current radio system, the maximum reliable distance is around 4.75m for 10dBm, 6.75m for 20dBm and 7.5m for 30dBm.

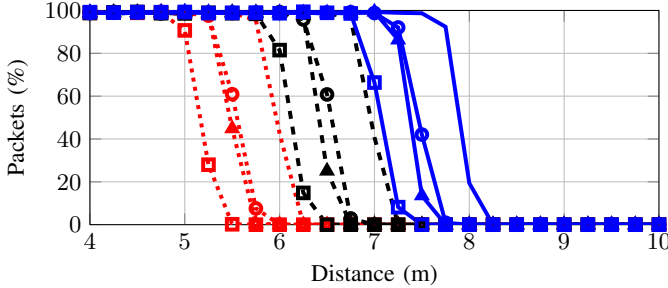


Fig. 5. Successfully received packets vs. distance (m).

B. Network routing using unicast transmissions with ACK

In this scenario we simulate a complete network of 21 nodes, which are deployed over a field of 42×18 square meters in a 3 rows, 7 columns grid (see Figure 6). The main application is to collect data from the network by a sink.

We choose the Collection Tree Protocol (CTP) for routing packets to the sink [13]–[15]. CTP has been implemented in several operating system (e.g., TinyOS, Mantis OS, Contiki OS, Sun SPOTs). It is also offered by Castalia [16] as a set of routing and MAC modules. CTP uses 3 techniques to provide efficient and reliable routing: *i*) an accurate link estimator that uses feedback from both the data and

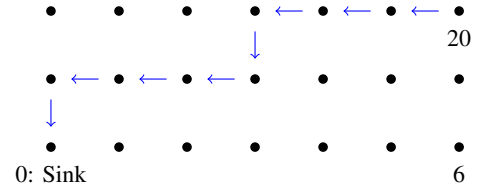


Fig. 6. Routing scenario.

control planes; *ii*) the Trickle algorithm [17] to time the control traffic, sending few beacons in stable topologies yet quickly adapting to changes; *iii*) actively probing the topology with data traffic, quickly discovering and fixing routing failures. CTP relies in acknowledged unicast transmissions for estimating the quality of the links.

When a node is chosen as parent from several neighbors, or it must perform many retransmissions attempts, its transmission queue can be fill up with unsent packets and further incoming packets are dropped. Since the dropped incoming packets are never acknowledged, the link seems down for its neighbors. This kind of congestion becomes worse when nodes operate at low data rates. Although CTP is able to detect and surmount congested nodes, there is some performance limit. We have studied the average number of successfully routed packets from any node to the sink as a function of the packet periodicity. Figure 7 shows that the application layer should wait at least 25 seconds between consecutive transmissions in order to successfully route 99.4% of the packets to the sink, and it must wait 40 seconds for routing 99.9% of the packets.

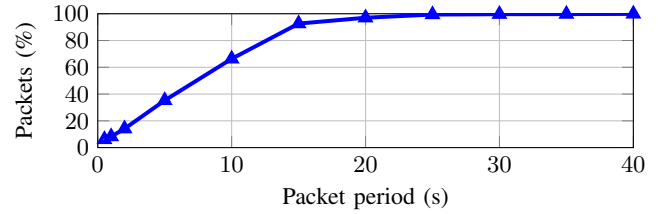


Fig. 7. Successfully routed packets from any node vs. packet period.

Another important metric is the time that takes a packet from any node to be routed to the sink. This latency depends on several issues, like how many hops are in the route, whether there are congested nodes that behave like bottlenecks, etc. Figure 8 shows a latency histogram. We have observed that, for this particular scenario, 95% of the received packets has latency lower than 2.5 seconds, 99% lower than 9 seconds and 99.8% lower than 25 seconds.

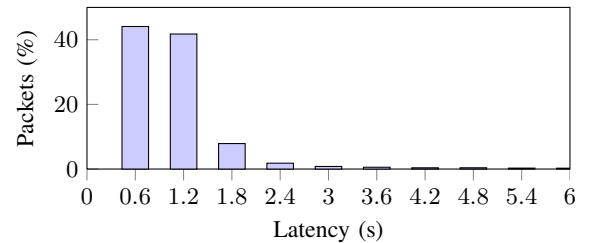


Fig. 8. Latency histogram for a packet to be received by the sink.

In order to perform these simulations, we extended the default `CtpTest` application such that the sink stores the packet number received and its source ID in a table. This way, we can discharge duplicated packets (due to retransmission) and know whether each data packet has been received by the sink. In addition, we increased the

ACK wait delay to 1.7 seconds (i.e., 5120 bits) in the **CC2420Mac** header file to work better with our low bit rate. We set the transmit power to 30dBm and used coherent receiver and channel coding, so that we achieved good connectivity and every node could find a route to the sink. Results have been averaged over 20 independent trials.

C. Diffusion unicast transmission with ACK

In this scenario, we consider the same network topology displayed in Figure 6. Nevertheless, instead of routing packets, here the nodes only communicate with their neighbors (see Figure 9), as if they were performing an iterative distributed algorithm (like those presented in companion paper [9]).

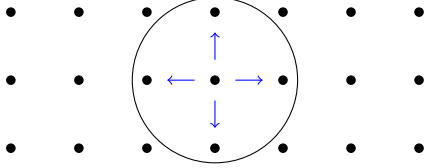


Fig. 9. Diffusion scenario.

Similar to Sec IV-B, we use acknowledged unicast transmissions. This means that every node has to figure out which nodes are its neighbors. CTP offers an efficient mechanism for neighborhood discovery. Hence, although we do not use routing in this scenario, we have extended CTP so that this neighborhood discovery mechanism can be used for diffusion too. This extension should be also useful for scenarios where nodes perform a distributed algorithm but also transmit data to the sink, combining autonomous and external monitoring. We followed these steps to extend CTP:

- 1) Create an extra field in the routing packet that accounts for the address of the destination neighbor.
- 2) Extend **CtpForwardingEngine::encapsulatePacket** to account for the new field.
- 3) Include in **CtpForwardingEngine::event_SubReceive_receive** the case where the node that receives the packet is the destination neighbor, so that it passes the packet to the application layer instead of forwarding it to its parent.

When using unicast transmissions, the nodes have to transmit and receive as many packets as neighbors per iteration. In order to minimize congestion, we have modeled an asynchronous algorithm in which, at every iteration, every node transmits its packet to all its neighbors, waiting some time between unicast transmissions (unicast period). Then, they wait some extra time—for allowing enough retries until the packets are acknowledged—before starting another iteration (contention period). Figure 2 shows the average percentage of successfully received packet vs. unicast period for different contention periods. We have observed that for unicast period 5 and contention period 100 (seconds) the received packet rate is greater than 95%, achieving 99% for 40 seconds of unicast period.

D. Diffusion broadcast transmission (no ACK)

In this section we also consider nodes transmitting to its neighbors over the same setting displayed by Figure 9. Nevertheless, here we suppose that the nodes broadcast only one packet for all its neighbors (instead of transmitting one packet per neighbor). This is achieved in Castalia by using the macro **BROADCAST_NETWORK_ADDRESS** as destination address. We study two cases: *synchronous* and *asynchronous* transmissions.

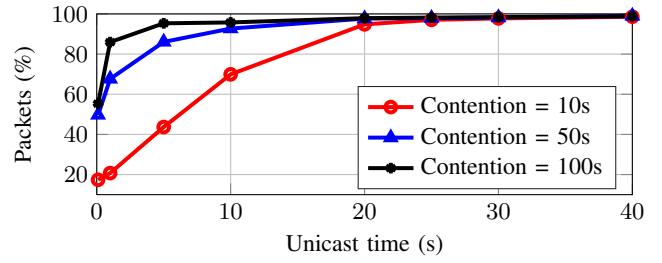


Fig. 10. Received packets vs. unicast period for multiple contention periods.

1) *Synchronous transmissions*: Every node transmits during its own time slot (i.e., TDMA). Hence, no packet is received while the node is transmitting, meaning that there are neither interference, nor collisions. The packet loss is only due to channel noise. In order to achieve this TDMA scheme we have used the application **connectivityMap**, setting the starting transmission time as follows:

$$t_i^s = PT_p Ni \quad (4)$$

where t_i^s denotes the starting time for node i , $P = 100$ is the number of packets (in a real algorithm this should be one, but here we want to average the channel noise), $T_p = 150\text{ms}$ is the wait time between consecutive packets from the same node, $N = 21$ is the number of nodes in the network and i is the node identity. Simulations show that we achieve 84.92% of successfully received packets. This is a relatively high percentage with minimum number of transmitted packets (neither ACK, nor retries are necessary), but the cost of achieving accurate TDMA synchronization should be taken into account when considering this option.

2) *Asynchronous transmissions*: There is no coordination among nodes. Hence, the packet loss is due to noise, interference and collisions. We have parametrized asynchronous transmissions by adding a term $\delta \in [0, 1]$ to (4):

$$t_i^a = \delta \cdot t_i^s \quad (5)$$

When $\delta = 0$, we have $t_i = 0$ for all $i = 1, \dots, N$ so that all nodes transmit at the same time (with exception of very small random clock drifts that can be dismissed for short simulation time). This makes all transmissions to collide since no node can receive any packet while it is busy with its own transmission. On the other hand, when $\delta = 1$, we recover the same result as for the perfect synchronization case studied in Sec. IV-D1. Figure 11 shows the average percentage of received packets vs. the overlapping parameter δ .

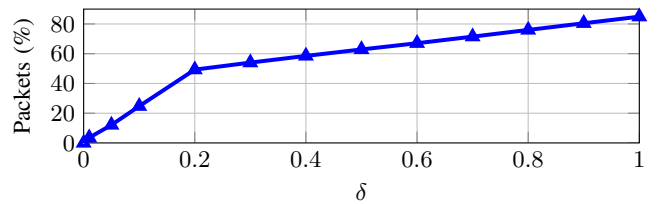


Fig. 11. Successfully received packets vs. asynchronous term δ .

V. CONCLUSION

We presented preliminary simulation results of U-WSN in Castalia, a realistic, flexible and fast simulator. We took real measurements of the underwater channel as starting point and proposed a simple linear path loss model. We considered routing as well as diffusion scenarios and presented some results that are useful for designing centralized

and distributed signal processing algorithms. In order to perform the simulations, we had to extend Castalia, implementing the proposed path loss model, including coherent receiver and channel coding to the radio and adding diffusion capabilities to CTP.

ACKNOWLEDGMENT

The authors would like to thank Juan Domingo Santana Urbin (ULPGC) for all the help provided while making the underwater measurements and to Gabriel Juanes and Raul Santana (PLOCAN) for setting up the measurement testbed in the pier and into the sea.

REFERENCES

- [1] P. Jepson, M. Arbelo, R. Deaville, I. Patterson, P. Castro, J. Baker, E. Degollada, H. Ross, P. Herráez, A. Pocknell *et al.*, “Gas-bubble lesions in stranded cetaceans,” *Nature*, vol. 425, no. 6958, pp. 575–576, 2003.
- [2] E. Parsons, S. J. Dolman, A. J. Wright, N. A. Rose, and W. Burns, “Navy sonar and cetaceans: Just how much does the gun need to smoke before we act?” *Marine Pollution Bulletin*, vol. 56, no. 7, pp. 1248–1257, 2008.
- [3] H. Wang, K. Yang, K. Zheng, Y. Han, and P. Xiao, “Experimental investigation on electromagnetic wave propagation across sea-to-air interface,” in *IEEE OCEANS*, Taipei, Taiwan, April 2014, pp. 1–6.
- [4] P. Mena, P. Dorta-Naranjo, G. Quintana, I. Pérez-Álvarez, E. Jiménez, S. Zazo, M. Pérez, L. Cardona, and J. Hernández Brito, “Experimental testbed for seawater channel characterization,” in *Submitted to Underwater Communications and Networking (UCOMMS2016)*, Lerici, Italy, September 2016.
- [5] E. Jimenez, G. Quintana, P. Mena, P. Dorta, I. Perez, E. Quevedo, and S. Zazo, “Investigation on radio wave propagation in shallow seawaters: simulations and measurements,” in *Submitted to Underwater Communications and Networking (UCOMMS2016)*, Lerici, Italy, September 2016.
- [6] P. Belanovic, S. Valcarcel Macua, and S. Zazo, “Distributed static linear gaussian models using consensus,” *Neural Networks*, vol. 34, pp. 96–105, 2012.
- [7] A. H. Sayed, S.-Y. Tu, J. Chen, X. Zhao, and Z. J. Towfic, “Diffusion strategies for adaptation and learning over networks,” *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 155–171, May 2013.
- [8] V. Savic, H. Wymeersch, and S. Zazo, “Belief consensus algorithms for fast distributed target tracking in wireless sensor networks,” *Signal Processing*, vol. 95, pp. 149–160, 2014.
- [9] J. Zazo, S. Zazo, S. Valcarcel Macua, M. Pérez Jiménez, I. Pérez-Álvarez, E. Jiménez, L. Cardona, and E. Quevedo, “Cooperative network node positioning techniques using underwater radio communications,” in *Submitted to Underwater Communications and Networking (UCOMMS2016)*, Lerici, Italy, September 2016.
- [10] D. Pediaditakis, Y. Tselishchev, and A. Boulis, “Performance and scalability evaluation of the castalia wireless sensor network simulator,” in *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010, p. 53.
- [11] A. Varga and R. Hornig, “An overview of the omnet++ simulation environment,” in *Proc. of the 1st Int. Conf. on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.
- [12] S. Lin and D. Costello, *Error Control Coding: Fundamentals and Applications*. Pearson-Prentice Hall, 2004.
- [13] O. Gnawali, R. Fonseca, K. Jamieson, M. Kazandjieva, D. Moss, and P. Levis, “CTP: An efficient, robust, and reliable collection tree protocol for wireless sensor networks,” *ACM Transactions on Sensor Networks (TOSN)*, vol. 10, no. 1, p. 16, 2013.
- [14] R. Fonseca, O. Gnawali, K. Jamieson, S. Kim, P. A. Levis, and A. Woo, “The collection tree protocol (CTP),” 2009, tinyOS TEP 123.
- [15] U. Colesanti and S. Santini, “A performance evaluation of the collection tree protocol based on its implementation for the castalia wireless sensor networks simulator,” ETH Zurich, Department of Computer Science, Technical Report, 2010.
- [16] —, “The collection tree protocol for the castalia wireless sensor networks simulator,” ETH Zurich, Department of Computer Science, Technical Report, 2011.
- [17] P. A. Levis, N. Patel, D. Culler, and S. Shenker, “Trickle: A self regulating algorithm for code propagation and maintenance in wireless sensor networks,” Computer Science Division, University of California, Technical Report, 2003.